```
<http://api.lodvader.aksw.org/distribution/compare/rdf/?retrieveDataset&source=http://dataset_example.org>
        a                       void:Linkset ;
        void:objectsTarget      <http://dataset_example.org#dataset> ;
        void:subjectsTarget     <http://dbpedia.org/dataid.ttl#article-categories_en> ;
        void:triples            "15" ;
        prov:wasGeneratedBy     <api.lodvader.aksw.org/distribution/compare/rdf/> .
```

Listing 1: void:linkset example

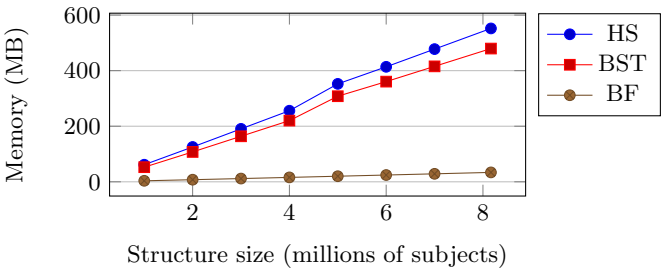

Figure 7: Memory usage per indexed resource

Parameter	Virtuoso	*LODVader*	Performance
Load time	24:02:36h	06:32:01h	3.67x faster
Disk usage	84,28Gb	4,03Gb	95.2% less pace

Table 1: Triple load time and disk usage of *LODVader* compared to Virtuoso. Virtuoso outperforms *LODVader*

queries is out of the scope of our approach. All experiments were made using a Intel(R) Core(TM) i7-5600U @ 2.6GHz, 16GB DDR3 and SSD drive. The results (Figure 7) show main advantages of using BF w.r.t. the memory usage while varying the number of resources for each structure. The difference from HS and BST to BF is notable. Storing 8 million resources HS, and BST use over 0.5 GB of RAM memory. Considering that a regular dataset can easily have more than this number of triples, the usage of HS and BST is unfeasible. It is important to stress that Figure 7 shows the memory usage for loading only one structure. However, usually a dataset is compared with not only one, but multiple datasets, and memory efficiency is fundamental when multiple BF are loaded at the same time. BF fulfills its function using less then 34MB of memory, performing on average 12 times better than HS and 10 times better than BST.

Apart from measuring the precision of BFs, we used OpenLink Virtuoso to compare the performance in other aspects. We are aware that *LODVader* is not a triplestore, and do not store sufficient data do make a SPARQL query. Parsing complexes queries is out of the scope of our approach. Moreover, considering the huge gain in the matter of performance of making simples searches, and storage space, *LODVader* is appropriate to index and search large amount of RDF data.

To compare the efficiency of *LODVader* with OpenLink Virtuoso, we loaded 491 datasets with a total of 1,092,182,333 triples. There is a slightly difference between the number of loaded triples on *LODVader* and Virtuoso. *LODVader* loaded 1,092,182,333 triples and Virtuoso 1,034,808,229. This difference is due the fact that *LODVader* stores triples regardless whether they are repeated or not, dealing with all triples as unique. On the other hand, triplestores will not store the same triple twice, considering that it's a graph structure. The problem is emphasized since each framework deals differently with erroneous data. For example, a bad RDF structure might be accepted by a particular framework, but completely ignored by others. Hence, when we deal with large amount of triples, it's difficult to have the exact number of resources in different frameworks.

Table 1 shows the results w.r.t for loading and indexing triples and the total space used in the hard drive. OpenLink Virtuoso loaded triples in 24:02:23, while *LODVader* 06:32:01, making the execution 3.67 times faster. The amount of data stored was 84,28Gb for OpenLink Virtuoso and 4,03Gb for *LODVader*.

5. ACKNOWLEDGEMENTS

This paper's research activities were funded by grants from the FP7 & H2020 EU projects ALIGNED (GA-644055), LIDER (GA-610782), FREME (GA-644771), Smart Data Web (GA-01MD15010B) and CAPES foundation (Ministry of Education of Brazil) for the given scholarship (13204/13-0).

6. REFERENCES

[1] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, July 1970.

[2] M. Brümmer, C. Baron, I. Ermilov, M. Freudenberg, D. Kontokostas, and S. Hellmann. DataID: Towards Semantically Rich Metadata for Complex Datasets. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 84–91. ACM, 2014.

[3] M. Grobe. RDF, Jena, SparQL and the 'Semantic Web'. In *Proceedings of the 37th Annual ACM SIGUCCS Fall Conference*, SIGUCCS '09, pages 131–138, New York, NY, USA, 2009. ACM.